Near Loop-free Routing: Increasing Path Choices with Stateful Forwarding

Klaus Schneider[1], Beichuan Zhang[1], Lan Wang[2], Lixia Zhang[3]
[1]The University of Arizona, [2]The University of Memphis, [3]UCLA
{klaus, bzhang}@cs.arizona.edu, lanwang@memphis.edu, lixia@cs.ucla.edu

ABSTRACT

When splitting traffic for one destination among multiple paths, the employed paths should be *loop-free*, lest they waste network resources, and the involved routers should be given a high *path choice*, that is, a high number of potential nexthops. In IP networks this requires the use of a loop-free routing protocol, which limits the achievable path choice.

Here we show that, in NDN, we can increase the path choice by combining a *Near Loop-free Routing* protocol (NLR) with on-demand loop removal at the forwarding layer. NLR routers 1) exclude the incoming face from forwarding, 2) use certain *heuristics* to minimize routing loops, and 3) remove any remaining loops at the forwarding plane. NLR achieves a higher path choice and path quality than current alternatives, while keeping computation complexity low.

ACM Reference format:
Klaus Schneider[1], Beichuan Zhang[1], Lan Wang[2], Lixia Zhang[3]. 2017. Near Loop-free Routing: Increasing Path Choices with Stateful Forwarding. In *Proceedings of ICN '17, Berlin, Germany, September 26–28, 2017,* 2 pages.
https://doi.org/10.1145/3125719.3132098

1 INTRODUCTION

IP routers cannot reliably detect loops, which makes loops *expensive*: packets circle around until the TTL runs out, wasting network resources while doing so. Thus, IP networks require a strictly *loop-free* routing protocol (see Section 2).

In contrast, Named Data Networking (NDN) can detect loops at the forwarding plane via a nonce in the packet header. Thus, NDN routing does *not* have to be loop-free, and indeed some NDN routing protocols [1] produce paths that may result in loops (Figure 1a). These loops, although detectable, waste network resources and thus need to be handled at the forwarding layer. This poses several questions: *1) Which routing protocol should we use to maximize the path choice while minimizing routing loops? 2) How should the forwarding layer deal with loops once detected? 3) And is the result better than using the best current loop-free routing protocol?*

For the case of multipath traffic splitting, current work does not answer these questions. Instead, current forwarding strategies avoid the looping problem in one of three ways: 1) they restrict themselves to forwarding on a single best path (NCC, BestRoute, and AccessStrategy in NFD, RGY [4], and ASF), 2) they split up traffic but rely on loop-free routing, 3) they ignore the problem and thus restrict their scalability (Multicast/Broadcast-Strategy in NFD).

Here we give a detailed answer to the first two questions, and answer the third one with a clear *Yes:* A combination of near-loopfree routing (NLR) and loop removal at the forwarding layer results in better performance than any reasonably complex loop-free routing protocol.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
ICN '17, September 26–28, 2017, Berlin, Germany
© 2017 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-5122-5/17/09…$15.00
https://doi.org/10.1145/3125719.3132098

2 LOOP-FREE ROUTING

To avoid loops, routers must use certain *heuristics* to decide which nexthops they use for forwarding. A simple heuristic is *Equal Cost Multi-Path* (ECMP), in which a router uses the shortest path nexthop n_{sp}, plus any nexthop n_i with the exact same cost: $cost(n_i, dst) = cost(n_{sp}, dst)$. However, ECMP only provides a small number of nexthops, since path costs need to match exactly. This nexthop choice can be extended by using the *Downward Path Criterion* (DW), which includes the shortest path nexthop plus any nexthop n_i that is closer to the destination than the current node (x): $cost(n_i, dst) < cost(x, dst)$. An even higher nexthop choice is achieved by MARA [2], which turns the network into Directed Acyclic Graphs (DAGs) and includes all links in the network.

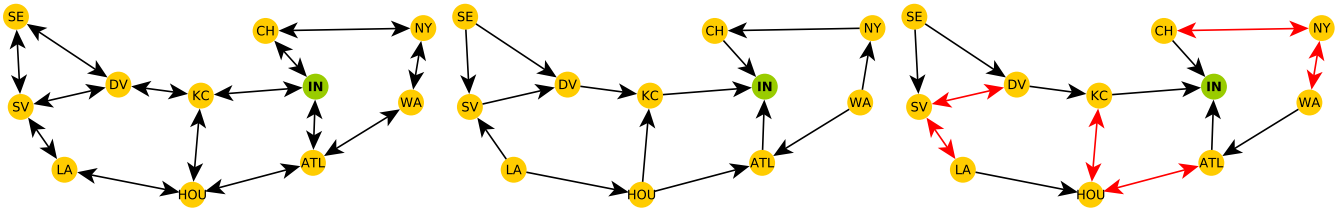
3 NEAR LOOP-FREE ROUTING

To increase path choice over the maximum that is possible in a DAG, NLR not only considers the destination prefix for forwarding, but also the *incoming interface*: a router always excludes the interface on which a packet arrived. This allows to use many more paths than otherwise possible. For example, Figure 1 shows the routes for the destination Indianapolis (IN) of the Abilene topology (viable nexthop entries are indicated by the direction of the arrows). In a DAG, NY only has one path to the destination: $NY \to CH \to IN$. With incoming-interface exclusion (NLR), NY can choose a second path: $NY \to WA \to ATL \to IN$. Figure 1 also shows the problem of a too broad heuristic (NLSR): It can lead to expensive loops like $KC \to DV \to SV \to LA \to HOU \to KC$.

NLR tries to find a heuristic broader than DAG but which still does not cause too many loops. For that goal, NLR assigns each nexthop a type based on its likelihood to loop:

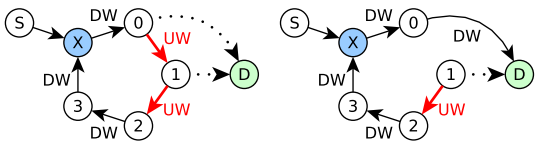
- `Downward:` The nexthop *never* causes a loop.
- `Upward:` The nexthop *may* cause a loop.
- `Disabled:` The nexthop *always* causes a loop.

Downward nexthops lead closer to the destination; upward nexthops lead further away. For brevity, we omit the details of the heuristics, which can be found in our longer report [3].



(a) NLSR (b) Loop-free Routing (DAG) (c) NLR

Figure 1: Routing Entries in the Abilene Topology for Destination Indianapolis (IN)



(a) Before (b) After

Figure 2: Example of Forwarding Loop Removal

4 FORWARDING LOOP REMOVAL

NLR may produce some looping paths, which need to be handled at the forwarding layer. After detecting a loop via a duplicate nonce, an NDN router can try different nexthops or backtrack to the previous hop, so that each Interest eventually reaches its destination. However, since this path exploration can be immensely costly, routers need a way to *permanently* avoid loops in the future; one of the routers involved in the loop needs to disable the nexthop it forwarded the packet on for all future packets towards the same destination. Thus, the question arises: *which nexthop should be disabled?*

Current NDN loop handling schemes disable the nexthop either at the router that detected the loop or at the router directly downstream to it (which is informed of the loop via a NACK) [4]. However, this will often disable nexthops that go closer to the destination, leading to higher forwarding overhead and longer remaining paths. We propose a more efficient solution called *Upward Nexthop Removal* (UNR): routers only remove (upward) nexthop entries that lead away from the destination; these nexthops may or may not be directly adjacent to the router detecting the loop. Moreover, a loop may contain multiple upward nexthops, and we need to disable only one of them to avoid future loops on this path. Thus, we remove the first upward nexthop *that the looped Interest encountered*, by introducing a special *loop signaling Interest* (see the rationale and design in [3]).

An example of the UNR algorithm is shown in Figure 2. The original Interest takes the path $S \to X \to 0 \to 1 \to 2 \to 3 \to X$, at which point X detects the loop by observing the duplicate nonce. A traditional loop-removal scheme would now disable either the nexthop $X \to 0$ or $3 \to X$, both of which are poor choices, as they lead closer to the destination. Instead, UNR sends the loop signaling Interest packet along the loop and router 0 will remove the upward nexthop $0 \to 1$. Afterwards, this loop is avoided, as node 0 will choose a different nexthop towards destination D.

Table 1: Routing Algorithms + Loop Removal

Routing	FW	NextHops (#)		NH>1	PathLen (Hops)		RemFibs (%)	
ECMP		1.39	(±0.86)	30.3	4.24	(±1.64)	0.00	(±0.00)
DW		2.57	(±2.41)	73.0	4.99	(±2.03)	0.00	(±0.00)
DWE		3.10	(±2.90)	83.5	5.91	(±2.47)	0.00	(±0.00)
MARA		3.10	(±1.46)	95.2	7.65	(±3.51)	0.00	(±0.00)
NLR	UNR	3.19	(±2.37)	99.6	6.54	(±2.85)	0.04	(±0.00)
NLSR2	UNR	1.85	(±0.35)	94.7	6.15	(±3.04)	2.50	(±0.02)
NLSR3	UNR	2.41	(±0.74)	93.9	6.14	(±2.96)	7.62	(±0.03)
NLSR	UNR	3.51	(±2.70)	95.6	6.49	(±2.91)	42.65	(±0.11)

5 EVALUATION

We compare NLR with the loop-free routing algorithms from Section 2, with NLSR [1], and with two NLSR variants that include the 2 or 3 lowest-cost nexthops (NLSR2/3). We have evaluated 9 different topologies [3], but for brevity we only show the results for the Rocketfuel Sprint network.

The results are shown in Table 1. ECMP only achieves a limited choice of nexthops, and in most cases restricts forwarding to only one nexthop (NH >1 shows the percentage of routers with more than one nexthops). DW and its extension DWE (explained in [3]) perform better than ECMP as they don't depend on exact cost matching. MARA includes the same average number of nexthops as DWE, which is the maximal possible number in a DAG. NLR outperforms all tested loop-free routing algorithms: in all tested topologies, it gives nodes both a higher number of average nexthops and a higher chance of having at least 2 nexthops. NLR achieves this with a low complexity of removing less than 1% of FIB entries (much lower than NLSR). The computation complexity of NLR falls in between the other schemes: it is more complex than DW and NLSR, but less complex than MARA (see [3]).

Moreover, NLR can handle 91.8% to 99.6% of link failures completely without invoking the routing protocol; routing convergence can be delayed or even completely avoided, if forwarding on slightly suboptimal paths is acceptable. Thus our work reduces the burden on routing protocols, allowing them to perform more time-intensive path computations than currently possible.

REFERENCES

[1] A. Hoque, S. O. Amin, A. Alyyan, B. Zhang, L. Zhang, and L. Wang. Nlsr: named-data link state routing protocol. In *ACM SIGCOMM ICN workshop*, 2013.
[2] Y. Ohara, S. Imahori, and R. Van Meter. Mara: Maximum alternative routing algorithm. In *INFOCOM 2009*. IEEE.
[3] K. Schneider and B. Zhang. How to establish loop-free multipath routes in named data networking. https://named-data.net/publications/techreports/ndn-0044-1-loopfree-routing/.
[4] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang. A case for stateful forwarding plane. *Elsevier*, 2013.